LANGUAGE MODELS CAN LEARN FROM VERBAL FEEDBACK WITHOUT SCALAR REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

LLMs are often trained with RL from human or AI feedback, yet such methods typically *compress nuanced feedback into scalar rewards*, discarding much of their richness and inducing scale imbalance. We propose treating verbal feedback as a conditioning signal. Inspired by language priors in text-to-image generation, which enable novel outputs from unseen prompts, we introduce the **feedback-conditional policy (FCP)**. FCP learns directly from response-feedback pairs, approximating the feedback-conditional posterior through maximum likelihood training on *offline* data. We further develop an *online bootstrapping* stage where the policy generates under positive conditions and receives fresh feedback to refine itself. This reframes feedback-driven learning as conditional generation rather than reward optimization, offering a more expressive way for LLMs to directly learn from verbal feedback.

1 INTRODUCTION

> *"That all of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)."* — Reward Hypothesis by Richard Sutton

The *reward hypothesis* in reinforcement learning (RL) was proposed over two decades ago (Sutton, 2004), when feedback from the environment had to be reduced to **scalar rewards** for RL algorithms to operate. This view shaped much of the field's progress and remains the prevailing standard in applying RL to *alignment* and *reasoning* for large language models (LLMs) (Ziegler et al., 2019; Bai et al., 2022; Rafailov et al., 2023; Guo et al., 2025).

Yet in practice, the feedback encountered in RL for LLMs, especially in *non-verifiable* settings, is most often **verbalized**, such as "`Good start, but the code can be more efficient`". Such feedback may come from human users (Stephan et al., 2024), generative reward models (Zhang et al., 2024; Mahan et al., 2024), or tool outputs in agentic scenarios (Wang et al., 2025b; Jin et al., 2025). Reducing the verbal feedback into scalar rewards introduces several limitations:

I. Information loss. Scalar rewards capture far less information than verbal feedback/critiques and are often uninterpretable. For example, the critiques "`The response is redundant but correct`" and "`The response is compact but has many typos`" may both collapse to a reward of 0.8, despite describing very different response patterns. Furthermore, the verbalized thoughts produced by (generative) reward models are typically discarded as intermediate outputs, with only the final scalar retained for RL training.

II. Ambiguity. Verbal feedback, especially from human users, is often *mixed* (containing both pros and cons), *emotional*, or *uncertain*, such as "`I'm so happy`" or "`I'm not sure, maybe try again?`". Such feedback is far more common than purely positive or negative signals and carries diverse cues for learning and for understanding user interaction styles. Mapping these forms of feedback to scalars could be unclear or arbitrary.

III. Imbalanced reward scales across tasks. In multi-task training (e.g., math, code, science, games), it is difficult to maintain a consistent reward scale. Positive feedback on a simple math problem is far easier to obtain than on a challenging coding or game-playing task, which induces imbalanced rewards across domains and biases the learning process.

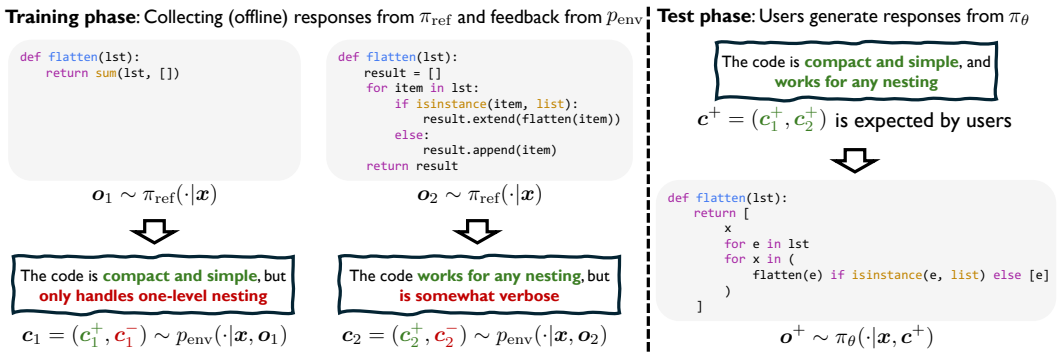


Figure 1: **Learning from mixed verbal feedback.** The instruction $\boldsymbol{x}$ is "*Write a Python function flatten(lst) that returns a flat list of integers*". The reference policy π_{ref} may assign low probability to the ideal response $\boldsymbol{o}^+$, making purely positive response-feedback pairs $(\boldsymbol{o}^+, \boldsymbol{c}^+)$ rare in the training data collected from π_{ref} and p_{env}. This resembles the setting of text-to-image generation, where the *language prior* enables models to combine seen captions (analogous to mixed feedback $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$) and generate rare images (analogous to purely positive response $\boldsymbol{o}^+$) such as "*a banana surfing on the ocean*" (Figure 4). Motivated by this, our model π_θ is trained as a feedback-conditional policy (FCP), and when conditioning on user-defined positive $\boldsymbol{c}^+$, there is $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$.

Scalarization has long been seen as unavoidable, bridging verbal feedback and the numerical signals required by RL. With the rise of large-scale language pretraining, however, this view is being re-examined (Yao, 2025). LLMs embody strong commonsense and linguistic **priors**, suggesting a new paradigm: *treat verbal feedback as a first-class training signal*, rather than forcing it into a scalar form.

After all, LLMs already show the ability to **implicitly understand verbal feedback**. In agentic tasks, they iteratively adapt by integrating feedback prompts from human users, external critiques, or tool calls into their context and refining their responses accordingly (Wang et al., 2025b; Novikov et al., 2025). This indicates that LLMs can process verbal feedback, but only implicitly, *through a latent "mental model" that does not convert understanding into explicit scalar rewards*. The key question, then, is how to distill such feedback into training so that it directly improves model performance, rather than relying on inefficient multi-turn trial and error at test time.

To this end, we propose to learn a **feedback-conditional policy (FCP)** $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$, where $\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})$ is a reference policy that generates a response $\boldsymbol{o}$ given an instruction $\boldsymbol{x}$, and $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$ is the distribution of environment feedback $\boldsymbol{c}$. Intuitively, the FCP reweighs the reference policy by how likely each response $\boldsymbol{o}$ would elicit the observed feedback $\boldsymbol{c}$. Conditioning on positive feedback $\boldsymbol{c}^+$ gives $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$, which increases the probability of generating responses that are more likely to receive favorable feedback. In this way, the FCP learns a *posterior* distribution that integrates prior knowledge from π_{ref} with verbal feedback, allowing it to handle diverse forms of feedback, including mixed ones, as illustrated in Figure 1.

After training an *offline* FCP $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$ that conditions on arbitrary feedback $\boldsymbol{c}$, we further improve it through *online bootstrapping*. Concretely, we conduct online training by sampling rollouts from the behavior policy $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$ (goal-conditioned on positive feedback), and re-annotating them with fresh feedback from p_{env}, thereby iteratively strengthening the policy.

Our pilot experiments show that FCP matches or surpasses strong scalar-based baselines such as offline RFT (Dong et al., 2023) and online GRPO (Shao et al., 2024), **without relying on verifiers, scalar conversion, or data filtering**. This demonstrates a simple and scalable framework that preserves the richness of verbal feedback while avoiding the scarcity of rule-based verifiers and the risk of reward hacking. While our current implementation is naive, advanced training techniques could further improve FCP's performance.

2 Learning Directly from Verbal Feedback

Traditional RL methods train a policy by up-weighting responses that receive "good" feedback and down-weighting those that receive "bad" feedback. From a probabilistic view, RL can be seen as learning a *posterior* over responses that are expected to receive good feedback (i.e., high rewards) (Peters & Schaal, 2007; Peng et al., 2019; Rafailov et al., 2023). Distinguishing what counts as good or bad typically requires carefully designed reward functions or detailed rubrics to produce scalar signals, leading to the limitations discussed in Section 1.

Our approach is inspired by language priors in text-to-image generation, where models compose *unseen* prompts from mixed captions (Figure 4). Similarly, language priors could enable LLMs to absorb diverse verbal feedback and yield high-quality responses beyond scalar reinforcement (Figure 1). Since LLMs already show implicit feedback understanding, we train directly on it: *offline* to initialize a **feedback-conditional policy (FCP)** (Section 2.1), then *online* to bootstrap performance (Section 2.2).

2.1 Offline Training: Initializing Feedback-Conditional Policy

We begin with a reference policy model π_{ref} that takes an input instruction $\boldsymbol{x}$ and generates a response $\boldsymbol{o} \sim \pi_{\text{ref}}(\cdot|\boldsymbol{x})$. The response $\boldsymbol{o}$ then undergoes a *single-turn* interaction with the environment, which provides verbal feedback $\boldsymbol{c} \sim p_{\text{env}}(\cdot|\boldsymbol{x}, \boldsymbol{o})$. The reference policy π_{ref} may represent a base model, an instruction-tuned model, or a reasoning model, and the response $\boldsymbol{o}$ can include both thinking processes and the final answer. The environment p_{env} may consist of human users or generative reward models. In the **offline** setting, where responses are collected from π_{ref}, we define the joint distribution of response-feedback pairs as $P_{\text{off}}(\boldsymbol{o}, \boldsymbol{c}|\boldsymbol{x}) \triangleq \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$, from which we derive the *feedback-conditional posterior* distribution:

$$P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}) = \frac{P_{\text{off}}(\boldsymbol{o}, \boldsymbol{c}|\boldsymbol{x})}{P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})} = \frac{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})}{\sum_{\boldsymbol{o}} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})}. \tag{1}$$

Informally, let $\boldsymbol{c}^+$ denote purely positive feedback and $\boldsymbol{c}^-$ purely negative one. Mixed feedback can be approximated as $\boldsymbol{c} = (\boldsymbol{c}^+, \boldsymbol{c}^-)$, while neutral or uncertain feedback may be neither. If we condition on positive feedback $\boldsymbol{c}^+$, for instance, "`The generated code is functionally correct, efficient, and compact`" for a coding instruction $\boldsymbol{x}$, then $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$, which favors responses $\boldsymbol{o}$ that are more likely to elicit positive feedback.[1]

While $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \propto \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$ appears to be the oracle policy we are seeking, it cannot be directly sampled from, because $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$ is defined only after the full response $\boldsymbol{o}$ is generated, and thus cannot guide generation step by step. We therefore aim to learn a policy that approximates $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$. Following Rafailov et al. (2023), we show that $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$ is the optimal solution to a KL-constrained reward maximization problem with reward function $\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$:

$$P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \in \arg\max_{\pi} \mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)} \left[\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})\right] - \mathbb{D}_{\text{KL}}\left(\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)||\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\right). \tag{2}$$

In the special case where the environment provides *verifiable rewards*, that is, $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o}^+) = 1$ for correct responses $\boldsymbol{o}^+$ and $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o}^-) = 0$ for incorrect responses $\boldsymbol{o}^-$, we can show that $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$ reduces to the optimal solution of a 0-1 binary reward maximization problem without KL regularization: $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+) \in \arg\max_{\pi} \mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)} \left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right]$ (proof is in Appendix A.1).

Alternative learning objective. In more general scenarios, particularly when feedback comes from human users, *solving Eq. (2) is typically intractable*. This is because we can only sample from p_{env} but cannot compute the exact log-likelihood $\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$. Note that the objective in Eq. (2) is equivalent to minimizing the *reverse* KL divergence between $\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$ and $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$:

$$\max_{\pi} \mathbb{E}_{\pi}\big[\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})\big] - \mathbb{D}_{\text{KL}}\big(\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)||\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\big) \Leftrightarrow \min_{\pi} \mathbb{D}_{\text{KL}}\big(\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)||P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)\big),$$

which is derived in Eq. (7). To avoid intractability of computing $\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$ in the reverse KL divergence, we instead turn to minimize the *forward* KL divergence between $\pi(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$ and $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$. This relaxation is standard in KL-constrained LM training, where forward-KL objectives have been used for distributional or reward-conditioned generation (Khalifa et al., 2020; Korbak et al., 2022; Pandey et al., 2024), though our setting differs in conditioning on rich verbal feedback. In practice, however, we can only obtain feedback from $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$, and it is infeasible to sample exclusively from the constrained subset of positive feedback $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x}, \boldsymbol{o})$ without carefully designed rubrics or filtering. To address this, we generalize the objective: rather than approximating only $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$, we learn to approximate $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})$, conditioning directly on *any* feedback $\boldsymbol{c}$.

Specifically, we propose to learn a **feedback-conditional policy (FCP)** $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})$ by minimizing the expected forward KL divergence between $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})$ and $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})$:

$$\begin{aligned}\min_{\pi_\theta} \mathbb{E}_{P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})} \left[\mathbb{D}_{\text{KL}}(P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})||\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}))\right] &\Leftrightarrow \max_{\pi_\theta} \mathbb{E}_{P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})} \left[\mathbb{E}_{P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})} \left[\log \pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})\right]\right] \\ &\Leftrightarrow \max_{\pi_\theta} \mathbb{E}_{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})} \left[\mathbb{E}_{p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})} \left[\log \pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})\right]\right],\end{aligned} \tag{3}$$

[1] Conditioning on negative feedback $\boldsymbol{c}^-$ would similarly favor poor responses, though this is rarely useful.

Algorithm 1 Offline training: Initializing feedback-conditional policy (Section 2.1)

Inputs: Reference policy $\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})$, feedback environment $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})$, feedback-conditional policy $\pi_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})$, instruction corpus $\mathcal{X}$, batch size B, optimizer $\mathcal{O}$

Outputs: The offline-trained parameters θ_{off}

1: **Initialize** $\pi_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}) = \pi_{\text{ref}}\left(\boldsymbol{o}\middle|\left[\texttt{<EF>}\boldsymbol{c}\texttt{</EF>},\boldsymbol{x}\right]\right)$, where <EF> and </EF> are special tokens used to wrap the expected feedback $\boldsymbol{c}$, which is concatenated before the instruction $\boldsymbol{x}$

2: **Collect** offline dataset $\mathcal{D}_{\text{off}} = \{(\boldsymbol{x},\boldsymbol{o},\boldsymbol{c})\}$ with $\boldsymbol{x}\sim\mathcal{X}$, $\boldsymbol{o}\sim\pi_{\text{ref}}(\cdot|\boldsymbol{x})$ then $\boldsymbol{c}\sim p_{\text{env}}(\cdot|\boldsymbol{x},\boldsymbol{o})$

3: **Objective:** $\max_\theta \mathbb{E}_{(\boldsymbol{x},\boldsymbol{o},\boldsymbol{c})\sim\mathcal{D}_{\text{off}}}[\log \pi_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})]$ // Taking expectation over $x\sim\mathcal{X}$ in Eq. (3)

4: **while** not converged **do**

5: Sample $\{(\boldsymbol{x}_i,\boldsymbol{o}_i,\boldsymbol{c}_i)\}_{i=1}^B \sim \mathcal{D}_{\text{off}}$; $\theta \leftarrow \mathcal{O}.\text{step}\big(\theta, \nabla_\theta \frac{1}{B}\sum_{i=1}^B \log \pi_\theta(\boldsymbol{o}_i|\boldsymbol{x}_i,\boldsymbol{c}_i)\big)$

6: **return** $\theta_{\text{off}} \leftarrow \theta$

where the second equivalence follows from the identities $P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})\cdot P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}) = P_{\text{off}}(\boldsymbol{o},\boldsymbol{c}|\boldsymbol{x}) = \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})$. This objective in Eq. (3) reduces to maximum likelihood training, which is straightforward to implement and optimize with data collected from $\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})$ and $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})$, as described in Algorithm 1. Its optimal solution is $\pi^*_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}) = P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})$ on the support set of $P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})$. Notably, our approach does not require explicitly distinguishing positive $\boldsymbol{c}^+$ from negative $\boldsymbol{c}^-$; the language prior embedded in LLMs can implicitly interpret and combine information from diverse forms of feedback, including mixed ones as seen in Figure 1. At test time, users may specify desired positive feedback $\boldsymbol{c}^+$, and responses can be generated from $\pi_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$.

Remark I: why using $P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})$? In Eq. (3), the expectation on $\boldsymbol{c}$ is taken w.r.t. $P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})$. In principle, any other distribution $p(\boldsymbol{c}|\boldsymbol{x})$ could be used, and the optimal solution $\pi^*_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}) = P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})$ would remain unchanged on the support $\text{supp}(p(\cdot|\boldsymbol{x}))$. We adopt $P_{\text{off}}(\boldsymbol{c}|\boldsymbol{x})$ mainly for two reasons: **(i)** its support set $\text{supp}(P_{\text{off}}(\cdot|\boldsymbol{x})) = \bigcup_{\boldsymbol{o}\in\text{supp}(\pi_{\text{ref}}(\cdot|\boldsymbol{x}))}\text{supp}(p_{\text{env}}(\cdot|\boldsymbol{x},\boldsymbol{o}))$ covers all feedback that may be encountered when collecting offline data; **(ii)** it serves as a compensating distribution that converts the intractable posterior expectation $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})$ into the tractable joint expectation $P_{\text{off}}(\boldsymbol{o},\boldsymbol{c}|\boldsymbol{x}) = \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})$, which is convenient to sample from.

Remark II: FCP as inverse dynamics. We observe that our FCP learning in Eq. (3) aligns with modeling *inverse dynamics* (Brandfonbrener et al., 2023), complementing supervised finetuning (SFT) as *behavior cloning*, and critique finetuning (CFT) (Wang et al., 2025a) as *forward dynamics*. A detailed discussion of this analogy is provided in Appendix A.2.

2.2 Online Training: Bootstrapping by Conditioning on Positive Feedback

We denote the model obtained by solving the offline problem in Eq. (3) as $\pi_{\theta_{\text{off}}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})$, which is capable of generating responses conditioned on any user-defined feedback $\boldsymbol{c}$. Building on this model, we further perform **online training** to bootstrap performance by *conditioning explicitly on positive feedback* $\boldsymbol{c}^+$. Concretely, we iteratively update parameters θ_{t+1} using rollouts from $\pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$ for $t\in\mathbb{N}$, with $\theta_0 = \theta_{\text{off}}$ initialized from the offline solution, as described in Algorithm 2.

Formally, we define the joint distribution $P_{\theta_t}(\boldsymbol{o},\boldsymbol{c},\boldsymbol{c}^+|\boldsymbol{x}) \triangleq p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})\cdot\pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)\cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})$, where $p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})$ denotes the distribution (fixed or trainable) of user-specified *expected* positive feedback. The corresponding feedback-conditional posterior is

$$P_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}) = \frac{P_{\theta_t}(\boldsymbol{o},\boldsymbol{c}|\boldsymbol{x})}{P_{\theta_t}(\boldsymbol{c}|\boldsymbol{x})} = \frac{\sum_{\boldsymbol{c}^+} p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})\cdot\pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)\cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})}{\sum_{\boldsymbol{o}}\sum_{\boldsymbol{c}^+} p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})\cdot\pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)\cdot p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})}. \tag{4}$$

The optimization objective for updating θ_{t+1} based on θ_t (with gradients stopped through θ_t) is

$$\begin{aligned}
&\min_{\pi_{\theta_{t+1}}} \mathbb{E}_{P_{\theta_t}(\boldsymbol{c}|\boldsymbol{x})}\left[\mathbb{D}_{\text{KL}}(P_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})||\pi_{\theta_{t+1}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}))\right] \\
\Leftrightarrow &\max_{\pi_{\theta_{t+1}}} \mathbb{E}_{P_{\theta_t}(\boldsymbol{c}|\boldsymbol{x})}\left[\mathbb{E}_{P_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})}\left[\log \pi_{\theta_{t+1}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})\right]\right] \\
\Leftrightarrow &\max_{\pi_{\theta_{t+1}}} \mathbb{E}_{p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})}\left[\mathbb{E}_{\pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{E}_{p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})}\left[\log \pi_{\theta_{t+1}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})\right]\right]\right].
\end{aligned} \tag{5}$$

Intuition. In each training round t (distinct from the s-th gradient steps taken within a round), the current model π_{θ_t} is conditioned on $\boldsymbol{c}^+$ to sample candidate positive responses. These responses are

Algorithm 2 Online training: Bootstrapping by conditioning on positive feedback (Section 2.2)

Inputs: Initialize $\theta_0 = \theta_{\text{off}}$ from Algorithm 1, user-desired feedback $p_{\text{user}}(\boldsymbol{c}^+|\boldsymbol{x})$, environment $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$, instruct. corpus $\mathcal{X}$, training rounds T, steps per round S, batch size B, optimizer $\mathcal{O}$
Outputs: The online-bootstrapped parameters θ_T

for $t = 1$ **to** T **do**
  $\theta_t \leftarrow \theta_{t-1}$
  for all instructions $\boldsymbol{x} \sim \mathcal{X}$ sampled in this round **do**
    Rollout $\boldsymbol{c}^+ \sim p_{\text{user}}(\cdot|\boldsymbol{x})$, $\boldsymbol{o} \sim \pi_{\theta_{t-1}}(\cdot|\boldsymbol{x}, \boldsymbol{c}^+)$ then obtain fresh critique $\boldsymbol{c} \sim p_{\text{env}}(\cdot|\boldsymbol{x}, \boldsymbol{o})$
    Push $(\boldsymbol{x}, \boldsymbol{o}, \boldsymbol{c})$ to buffer $\mathcal{B}^t_{\text{on}}$ // $\boldsymbol{c}$ is usually different (at least linguistically) from $\boldsymbol{c}^+$
  Objective: $\max_{\theta_t} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{o},\boldsymbol{c})\sim\mathcal{B}^t_{\text{on}}}[\log \pi_{\theta_t}(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c})]$ // Taking expectation over $\boldsymbol{x} \sim \mathcal{X}$ in Eq. (5)
  for $s = 1$ **to** S **do**
    Sample $\{(\boldsymbol{x}_i, \boldsymbol{o}_i, \boldsymbol{c}_i)\}_{i=1}^B \sim \mathcal{B}^t_{\text{on}}$; $\theta_t \leftarrow \mathcal{O}.\text{step}\big(\theta_t, \nabla_{\theta_t} \frac{1}{B} \sum_{i=1}^B \log \pi_{\theta_t}(\boldsymbol{o}_i|\boldsymbol{x}_i, \boldsymbol{c}_i)\big)$
return θ_T

then re-annotated with fresh feedback $\boldsymbol{c}$ from the environment. Over successive rounds, the model learns to identify cases where conditioning on $\boldsymbol{c}^+$ does not in fact yield positive critiques, while reinforcing those that align with the expected feedback. This iterative process bootstraps the model, progressively strengthening alignment with user-specified positive feedback. Moreover, following Lanchantin et al. (2025), the number of gradient steps S between rounds can be flexibly adjusted, allowing the procedure to interpolate between fully online and semi-online training.

Test-time usage. At test time, FCP does not require any environment-generated feedback. If the user wishes, they may provide a desired feedback condition $\boldsymbol{c}_{\text{user}}$. When no such condition is given, we automatically obtain a positive feedback description by prompting an LLM with instructions such as: "`Provide several possible positive feedback descriptions for the following query.`" This produces a suitable $\boldsymbol{c}^+$ for conditioning. The model then generates an output $\boldsymbol{o}$ using the feedback conditional policy $\pi_\theta(\boldsymbol{o}|\boldsymbol{x}, \boldsymbol{c}^+)$. This is a single-pass generation process that does not require any iterative refinement. In practice, inference simply consists of prepending the selected feedback description to the input prompt.

3 Experiments

We evaluate FCP on mathematical and general reasoning tasks, aiming for a direct comparison with scalar-based methods. We choose reasoning tasks as the testbed because scalarized RL has been especially successful in this domain (Guo et al., 2025; Ma et al., 2025), making it a strong and convincing benchmark. Showing that FCP performs comparably under such demanding conditions provides a rigorous test of its effectiveness. As shown in Section 3.2, FCP indeed matches scalar pipelines, with more design choices presented in our ablation studies (Section 4).

3.1 Setup

Datasets and models. For mathematical reasoning, we use `Big-Math` (Albalak et al., 2025), a 251k-problem dataset curated for training and evaluation. For general reasoning, we use `WebInstruct` (Yue et al., 2024) from General-Reasoner (Ma et al., 2025). Its multi-domain, free-form answers are unsuitable for rule-based filters, so prior work relies on *generative reward model*—making it a natural testbed to contrast verbal conditioning with scalar-reward pipelines. As pilot experiments, our base model is `Qwen2.5-7B-base` (Yang et al., 2024).

Feedback environment simulation. Human feedback is costly and difficult to standardize in both quality and style. We therefore simulate the feedback environment with `GPT-5-nano`, which provides feedback for both offline (Algorithm 1) and online (Algorithm 2) training. Our method only requires feedback to be *non-deceptive* (following p_{env}), rather than a detailed breakdown, making lightweight models sufficient. To implement this, we design a unified prompt template (Figure 6) that first elicits a low-quality, *real-world user*-style feedback, then a high-quality, *professional reviewer*-style feedback covering multiple aspects, and finally a *scalar score* summarizing overall quality. This setup ensures that the same feedback source supplies both the *verbal conditions* for FCP and the *scalar rewards* for RL baselines, enabling a fair comparison.

Table 1: **Math (in-domain) and IFEval (out-of-distribution) results.** Here **Avg.** denotes mean accuracy (%) over five math benchmarks. CFT is critique finetuning (Wang et al., 2025a), see Section 4.2, and Critique-GRPO is adopted from Zhang et al. (2025).

Offline Algo. + Online Algo.	Scalar Reward	In-Domain AIME24	AIME25	MATH500	Minerva	Olympiad	Avg.	OOD IFEval
Base	-	7.5 ±1.7	6.7 ±0.0	63.8 ±63.8	28.3 ±0.8	28.6 ±0.4	27.0 ±0.5	36.8
+ GRPO	✓	20.0 ±0.0	**13.3** ±2.7	75.7 ±1.7	42.3 ±1.3	**40.8** ±0.5	38.4 ±0.9	38.5
+ Critique-GRPO	✓	15.0 ±1.9	9.2 ±1.7	**76.8** ±0.3	36.1 ±0.9	40.1 ±0.5	35.4 ±0.6	**39.0**
RFT	✓	13.3 ±0.0	3.3 ±0.0	69.2 ±0.6	32.4 ±0.6	33.8 ±0.9	30.4 ±0.0	37.5
+ GRPO	✓	**25.8** ±1.7	9.2 ±1.7	75.1 ±0.7	36.8 ±0.9	38.9 ±0.1	37.1 ±0.5	38.8
+ Critique-GRPO	✓	16.7 ±4.7	9.2 ±5.0	75.2 ±0.4	35.8 ±0.7	39.6 ±0.5	35.3 ±1.2	38.6
CFT	✗	1.7 ±3.3	0.0 ±0.0	27.0 ±3.2	9.2 ±6.4	7.7 ±1.4	9.1 ±1.3	-
FCP	✗	6.7 ±0.0	3.3 ±3.8	68.9 ±1.0	31.2 ±1.1	32.4 ±1.1	28.5 ±1.1	38.6
+ Bootstrap	✗	25.0 ±3.3	7.5 ±1.7	76.5 ±0.7	**45.8** ±0.7	38.8 ±0.6	**38.7** ±0.7	**39.0**

Baselines. We compare against two strong baselines: Rejection Sampling Finetuning (RFT) and GRPO (Dong et al., 2023; Shao et al., 2024). RFT filters responses by correctness and finetunes only on the correct ones, which in the *offline* case reduces to training on a binary scalar score (correct/incorrect). While simple and effective, it depends on reliable filtering and a stable verifier. GRPO instead uses group-normalized scalar rewards to estimate advantages and has become one of the strongest *online* methods, especially in math reasoning where answers can usually be verified automatically. Both baselines rely on scalar-based filtering or scoring, making them dependent on high-quality verifiable data and an auxiliary verifier. Even rubric-based reward shaping (Zhou et al., 2025b) still loses much of the feedback richness. Our experiments thus offer a stringent comparison between scalar-reward pipelines (RFT/GRPO) and FCP learning.

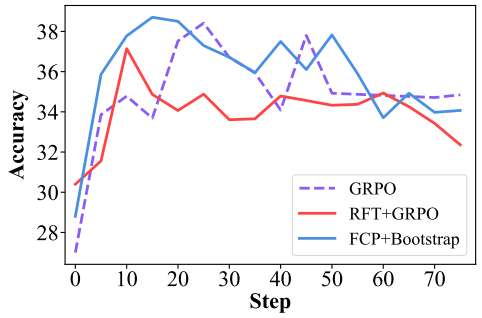
Training details for FCP. In the *offline* stage (Algorithm 1), the base model generates 8 candidate responses per prompt. We discard prompts where all responses are entirely correct or incorrect, then sample one correct and one incorrect response for `GPT-5-nano` to provide feedback. All collected feedback is used to train FCP, while a pool of positive feedback $\{\boldsymbol{c}^+\}$ is built from the scalar scores in the feedback. In the *online* stage (Algorithm 2), for each prompt $\boldsymbol{x}$ we sample a desired condition $\boldsymbol{c}^+ \sim p_{\text{user}}(\cdot|\boldsymbol{x})$ by drawing from the pool $\{\boldsymbol{c}^+\}$. For rollout, the prompt batch size is 2048 with 4 responses per prompt; for training, the mini-batch size is 512, giving 4 gradient updates per rollout step. Each response receives a fresh *professional reviewer*-style feedback from `GPT-5-nano`, which is concatenated with the prompt and response (using the Algorithm 1 wrapper `<EF>` and `</EF>`) for cross-entropy training. This bootstrapping loop improves response quality under desired conditions while grounding updates in new feedback. For fair comparison, GRPO is trained with the same scalar scores from `GPT-5-nano` under the identical prompt template.

Evaluation. We assess mathematical reasoning on AIME24&25, MATH500 (Hendrycks et al., 2021), Minerva-Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024), and general reasoning on GPQA-Diamond (Rein et al., 2024), MMLU-Pro (Wang et al., 2024), and TheoremQA (Chen et al., 2023). To test instruction-following beyond the training domain, we also include IFEval (Zhou et al., 2023). All benchmarks use a unified protocol: each dataset is run under four random seeds [2], with mean accuracy reported. Inference uses `vllm` (Kwon et al., 2023) with greedy decoding and a maximum generation length of 8192 tokens. For FCP, we match the training setup by randomly sampling one feedback condition from $\{\boldsymbol{c}^+\}$ for each question and prepending it to the prompt template.

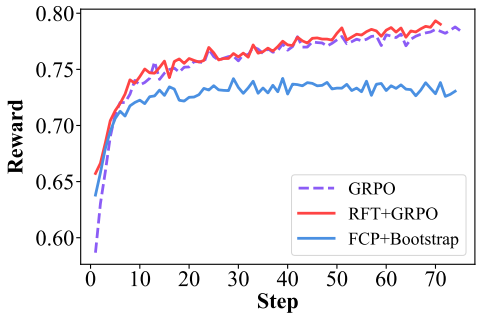
3.2 Main Results

Offline FCP is comparable to RFT. On `Qwen2.5-7B-base`, offline FCP attains 28.8% average accuracy on the math suite, between the base model (27.0%) and RFT (30.4%) (Table 1). General reasoning shows the same order: 38.7%, 43.5%, and 44.6% for base, FCP, and RFT (Table 2). This is expected, since FCP directly learns from all response-feedback pairs without filtering and therefore inevitably absorbs noise, whereas RFT benefits from elaborate correctness filtering. Still, FCP remains competitive under noisier supervision.

[2] For MMLU-Pro, the large test set size (∼12k questions) simultaneously leads to a high evaluation cost and an inherently small variance (typically ≤ 0.3). Consequently, we report results based on a single evaluation run for this benchmark.



(a) Average accuracy over five math benchmarks measured at intermediate checkpoints.



(b) Scalar scores assigned by `GPT-5-nano` to model rollouts during training.

Figure 2: **Training dynamics of FCP and scalar-based baselines.** (a) FCP+Bootstrap matches GRPO and RFT+GRPO accuracy within 30 steps. (b) In contrast, its scalar reward scores lag behind, consistent with the fact that FCP does not directly optimize against reward model's preference.

Table 2: **General reasoning results.** Accuracy (%) across three benchmarks and their average.

Offline Algo. + Online Algo.	Scalar Reward	GPQA-Diamond	MMLU-Pro	TheoremQA	Average
Base	-	27.9 ±1.0	49.7	38.6 ±0.2	38.7 ±0.4
+ GRPO	✓	32.5 ±5.3	49.7	**49.4** ±1.4	43.9 ±1.7
RFT	✓	35.2 ±1.3	55.0	43.7 ±0.9	44.6 ±0.2
+ GRPO	✓	37.2 ±2.5	**57.0**	48.3 ±0.2	47.5 ±0.8
FCP	✗	35.0 ±2.9	53.6	42.0 ±1.0	43.5 ±0.6
+ Bootstrap	✗	**39.1** ±2.9	55.3	49.1 ±0.5	**47.8** ±0.9

Bootstrapping enables FCP to rival scalarized RL baselines. Online bootstrapping lifts FCP from 28.8% to 38.7% average accuracy on the math suite (Table 1), slightly surpassing GRPO (38.4%). A similar trend appears in out-of-distribution case: on **IFEval**, FCP+Bootstrap reaches 39.0%, comparable to GRPO (38.5%) and RFT+GRPO (38.8%). General reasoning benchmarks (Table 2) show the same pattern, with FCP+Bootstrap at 47.8%, matching the best scalar-based baseline (47.5%). These results indicate that bootstrapping gives FCP the effectiveness of scalarized RL while retaining the advantage of learning directly from richer verbal feedback.

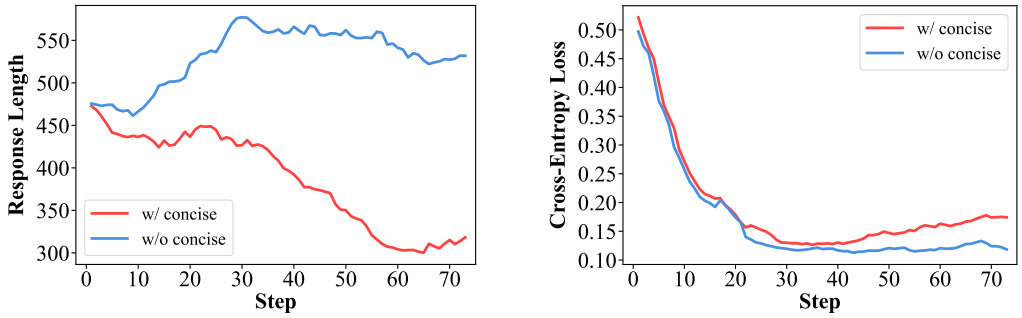
3.3 Learning dynamics of FCP

FCP enables controllable behavior across diverse feedback conditions. A core question is whether the policy truly *learns* the conditioning signal $\boldsymbol{c}$—and, if so, whether this lets us absorb negative samples into training without hurting best-case performance. We probe this by sampling representative feedback from the offline pool and evaluating under several conditions.

Table 3 shows a sharp contrast on MATH500: accuracy is 68.5% under `fully_positive` but only 17.1% under `fully_negative`, far below the base model's 63.8% (Table 1). This indicates the model internalizes the control signal: negative conditions induce poor behavior when requested, yet positive conditions still yield strong accuracy—showing that including negative samples in training (*using the same cross-entropy loss as positives*) does not cap performance under positive ones.

Other conditions also shift behavior as intended. Under `neutral`, where the condition $\boldsymbol{c}$ asks for a correct answer and a more verbose solution, accuracy drops slightly but response length grows, reflecting a trade-off. With `has_code`, the share of responses containing code rises to 74.3%, confirming that stylistic attributes in $\boldsymbol{c}$ are also followed. Compared to `Qwen2.5-7B-Instruct`, which shows little variation across conditions due to training only on verified positives, FCP learns to map feedback $\boldsymbol{c}$ to distinct behaviors, enabling broad data use without manual filtering.

FCP achieves strong accuracy without over-optimizing scalar rewards. As seen in Figure 2a, both FCP and GRPO reach peak accuracy within 30 online steps, with scalar scores from `GPT-5-nano` rising sharply at the start. Yet Figure 2b shows FCP's scores lagging behind GRPO's later, since it does not directly optimize against the scalar reward model. Crucially, FCP sustains high accuracy



(a) Average response length over training steps. (b) Cross-entropy loss over training steps.

Figure 3: **Effect of length-related conditions on bootstrapping stability.** Both curves are smoothed with a 10-step moving average. (a) Without filtering, response length decreases over time, while filtering out length-related conditions leads to steady growth. (b) The corresponding loss curves show greater instability when length-related conditions are included.

Table 3: **Comparison under different feedback conditions.** Accuracy (%), code ratio (proportion of responses containing code), and average response length are all measured on MATH500.[3]

		Example 1	Example 2	Example 3	Example 4
Feedback type		`fully_positive`	`fully_negative`	`neutral`	`has_code`
Content		Accurate and clear; concise and coherent reasoning; correct conclusion.	Incoherent and incomplete. Random and unfocused. Unclear and disorganized.	Correct and readable overall, but the solution is **verbose** and could be streamlined for tighter logical flow.	Correct and clear, though slightly verbose with superfluous **code**.
Accuracy	Instruct	76.2	77.4	77.5	76.6
	FCP	68.5	**17.1**	61.1	53.9
Code Ratio	Instruct	0	0	0	0
	FCP	22.7	55.6	46.3	**74.3**
Response Length	Instruct	632	650	638	661
	FCP	605	1442	**722**	659

despite lower scores, indicating it avoids the reward-hacking behavior often seen in scalar-based methods and underscoring verbal feedback as a more robust training signal.

Length-related conditions destabilize FCP bootstrapping. We find that feedback conditions c^+ tied to output length, such as *conciseness*, can destabilize online bootstrapping. As shown in Figure 3, these conditions cause average response length to shrink over time while the loss becomes unstable. This likely reflects a feedback loop: concise rollouts receive affirming feedback, and cross-entropy updates further shorten responses, eventually collapsing output length. Filtering out length-related conditions instead yields steadily longer responses, mirroring GRPO's training behavior (Guo et al., 2025) and supporting the view that reliable math solving benefits from extended reasoning traces.

4 Ablation Studies

Unless otherwise noted, we use the following *default* configuration: For rollout, the prompt batch size is 512 with 4 responses generated per prompt. For training, the mini-batch size is 512, corresponding to a single gradient update per rollout step, which yields a fully online setting. All rollouts of the same prompt share an identical feedback condition c^+. Training uses token-level mean loss aggregation, with fresh feedback c provided in the professional *reviewer*-style by `GPT-5-nano`.

4.1 Real-world user vs. professional reviewer style

Real-world user feedback is abundant and inexpensive but often noisy and inconsistent; professional reviewer feedback is higher quality but costly and less scalable. We therefore ask: *how much feedback quality does FCP actually require?* We use a unified prompt that asks `GPT-5-nano` to produce both a low-quality real-world *user*-style feedback and a high-quality professional *reviewer*-style feedback in a single response. As shown in Table 4, *user*-style feedback is typically subjective and colloquial, whereas *reviewer*-style feedback is precise and structured.

[3]For the Instruct model, evaluation prompts are wrapped as "`Your answer should be expected to get the following critique: <feedback_content>\n{question}`".

Table 4: Examples of feedback in *real-world user*-style and *professional reviewer*-style.

Role	Critique Type	Examples
Real-World User	`fully_positive`	That looks right to me, concise and easy to follow. I'm satisfied with the final result.
	`fully_negative`	I have no idea what you were trying to say—the response is nonsense and not helpful at all.
	`neutral`	I'm not completely sure about the logic, but the final answer matches the number I was expecting.
Professional Reviewer	`fully_positive`	Correct and clear; succinct and logically sound, with concise and effective reasoning.
	`fully_negative`	Incorrectly structured and incoherent. The reasoning is absent and the content is unusable.
	`neutral`	Correct final result but unclear and incomplete reasoning; concise yet insufficiently rigorous.

Table 5: Ablation results on hyperparameter choices, data sources, and feedback settings. Reported numbers are average accuracy on math benchmarks; Δ shows change relative to the default setting.

Variant	Changed Setting(s)	Avg Acc	Δ
Default	——	35.3	0.0
w/ user style feedback	`critique_type`=user	32.8	-2.5
w/ partial online	`prompt_bsz`=2048	38.7	+3.4
w/ unbiased loss	`loss_agg_mode`=seq-mean-token-sum	36.0	+0.7
w/ smaller batch size	`train_bsz`=`ppo_mini_bsz`=256	37.7	+2.4
w/ more diverse $\boldsymbol{c}^+$	use random $\boldsymbol{c}^+$ per rollout	34.1	-1.2
w/ different dataset	use MATH-Train split	34.3	-1.0

Table 5 shows that using only *user*-style feedback (offline and online) lowers math-suite accuracy by 2.5 points relative to *reviewer*-style feedback, yet still delivers a +5.8 gain over the base model (27%; Table 1). While *reviewer*-style feedback is more effective, *user*-style feedback remains surprisingly competitive after FCP training. Its lower cost and broad availability make it a practical source for scaling, with *reviewer*-style feedback reserved for targeted quality improvements.

4.2 Additional training design choices and comparison to CFT

We further study how different design choices affect FCP training, with results summarized in Table 5.

Online update strategy. Compared to the fully online setup, using a larger prompt batch size of 2048 while keeping the mini-batch size fixed at 512 results in four gradient updates per rollout step, and yields better accuracy. This suggests that partial online updates can improve optimization efficiency.

Loss aggregation. In Algorithm 2, cross-entropy on self-sampled responses reduces to policy gradient with unit advantages, which suffers from length bias (Liu et al., 2025a). A debiased scheme averaging at the sequence level and summing at the token level gives a consistent +0.7% gain.

Other variations. We also experimented with several alternative configurations. Reducing the training batch size to 256 improves accuracy by about +2.4%. Training the online stage on a dataset different from that used for offline pretraining slightly underperforms the default baseline, yet remains +5.5% above the offline-only initialization, indicating that offline and online datasets need not be strictly aligned for FCP to be effective.

Comparison to Critique Finetuning (CFT). CFT can perform well with high-quality and detailed critiques (Wang et al., 2025a), but applying it to the same coarse and lightweight feedback used for FCP leads to severe degradation—worse than the base model (Table 1). This highlights a key strength of FCP: it effectively leverages coarse, high-level feedback without costly fine-grained annotations.

5 Related work

SFT and RL methods for reasoning. The ability to perform reasoning has become a defining strength of LLMs, enabling progress across mathematics, coding, and scientific domains (Jaech et al., 2024; Comanici et al., 2025). To enhance these skills, two approaches have proven especially influential: SFT and RL (Uesato et al., 2022; Rafailov et al., 2023; Guha et al., 2025; Hu et al.,

2025; Hochlehnert et al., 2025). Following the success of the DeepSeek-R1 recipe (Shao et al., 2024; Guo et al., 2025), a number of RL variants have been introduced, including Dr. GRPO (Liu et al., 2025a), DAPO (Yu et al., 2025), REINFORCE++ (Hu, 2025), and VAPO (Yue et al., 2025). Beyond algorithmic proposals, researchers have systematically investigated the RL design space for reasoning (Zeng et al., 2025; Team et al., 2025), examining factors such as staged training curricula (Wen et al., 2025; Luo et al., 2025) and reward formulation (Gao et al., 2024; Cui et al., 2025; Qi et al., 2025; Zhou et al., 2025a). While much of the initial progress focused on mathematics, these methods have more recently been extended to software engineering and code reasoning (Liu & Zhang, 2025; Xie et al., 2025; Wei et al., 2025; Yang et al., 2025; Chen et al., 2025), as well as to broader agentic applications (Wang et al., 2025b; Jin et al., 2025; Jiang et al., 2025; Xue et al., 2025).

Learning from verbal feedback. Most existing approaches convert verbal feedback into scalar rewards for RL training (Kim et al., 2024; Ankner et al., 2024; Lightman et al., 2024; Stephan et al., 2024; Whitehouse et al., 2025; Liu et al., 2025b). More recent efforts explore learning directly from feedback or critiques: Lloret et al. (2024) propose conditional SFT based on toxicity categorization in alignment tasks, CFT (Wang et al., 2025a) trains models to imitate critiques, Critique-GRPO (Zhang et al., 2025) incorporates critique-guided refinements into online RL, Salemi & Zamani (2025) jointly optimize a feedback model and a policy model, and Chen et al. (2024) introduce a refinement model that corrects errors using feedback. These approaches generally assume feedback is high-quality, informative or categorized, and reliably improves self-refinement. In practice, however, human feedback is often mixed, free-form, emotional, or uncertain. Moreover, while such feedback is easy to collect, its distribution is difficult to model with generative reward models that must capture diverse user interaction styles. In contrast, our FCP framework does not require feedback to be high-quality or rubric-constrained; by treating feedback as a conditioning signal rather than a prediction target, it can flexibly exploit the full range of verbal feedback, including noisy or mixed forms, for training.

6 Discussion and Future Directions

Our key insight is that the essence of RL lies in *online interaction with the environment*, not in scalar rewards or any specific algorithm. Scalarization was historically necessary for control-centric RL in robotics or strategy-centric RL in games, but it may not be intrinsic to language-centric systems like LLMs. This reopens the debate around the reward hypothesis: earlier critics could only offer counterexamples without an alternative framework (Skalse & Abate, 2022), whereas our FCP approach leverages **language priors** to provide a principled way to bypass scalar rewards. Crucially, during training, feedback $\boldsymbol{c}$ is a *dependent variable* generated from the environment $p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x}, \boldsymbol{o})$ and cannot be directly controlled, while at test time the conditioning feedback $\boldsymbol{c}^+$ becomes an *independent variable* freely specified by users. This asymmetry enables full use of diverse feedback during training while allowing precise controllability at inference. By directly mapping feedback to responses, our FCP bypasses reward imbalance, preserves feedback richness, and improves data efficiency. Unlike RFT (Dong et al., 2023; Touvron et al., 2023), which discards many useful data pairs, FCP retains diverse feedback, including mixed and uncertain, and can merge complementary signals across examples at test time (Figure 1). This establishes verbal feedback as a first-class training signal and FCP as a natural, scalable alternative to scalarized RL.

Future directions. Several extensions of FCP are promising. One is to *combine it with verifiable rewards*, for instance by treating the absence of feedback as a neutral condition (e.g., using the null feedback token `<EF></EF>`), so that reliable scalar supervision can complement verbal feedback when available. Another is to extend FCP to *multi-turn interactions*, where feedback is incorporated before the next turn of generation in a teacher-forcing style, enabling closer alignment with iterative human guidance. A third is *test-time adaptation*: by conditioning on a few user-provided examples, the model could rapidly adjust to individual feedback styles, similar to personalization in text-to-image generation. Finally, the feedback condition $\boldsymbol{c}$ could be made *multimodal*. Collectively, these future directions would deepen integration of natural feedback into LLM training, bridging offline and online stages while adapting to diverse user needs.

References

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and Nick Haber. Big-math: A large-

scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. Critique-out-loud reward models. *arXiv preprint arXiv:2408.11791*, 2024.

Sanjeev Arora, Simon Du, Sham Kakade, Yuping Luo, and Nikunj Saunshi. Provable representation learning for imitation learning via bi-level optimization. In *International Conference on Machine Learning*, pp. 367–376. PMLR, 2020.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. *Advances in Neural Information Processing Systems*, 36: 66953–66978, 2023.

Angelica Chen, Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Samuel R. Bowman, Kyunghyun Cho, and Ethan Perez. Learning from natural language feedback. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. *arXiv preprint arXiv:2305.12524*, 2023.

Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable rl with exogenous distractors via multistep inverse dynamics. *arXiv preprint arXiv:2110.08847*, 2021.

Jiaxuan Gao, Shusheng Xu, Wenjie Ye, Weilin Liu, Chuyi He, Wei Fu, Zhiyu Mei, Guangju Wang, and Yi Wu. On designing effective rl reward at training time for llm reasoning. *arXiv preprint arXiv:2410.15115*, 2024.

Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3), 2018.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*, 2025.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling reinforcement learning on the base model. https://github.com/Open-Reasoner-Zero/Open-Reasoner-Zero, 2025.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, Tianyu Pang, and Wenhu Chen. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. *arXiv preprint arXiv:2012.11635*, 2020.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, et al. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220, 2022.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Jack Lanchantin, Angelica Chen, Janice Lan, Xian Li, Swarnadeep Saha, Tianlu Wang, Jing Xu, Ping Yu, Weizhe Yuan, Jason E Weston, et al. Bridging offline and online reinforcement learning for llms. *arXiv preprint arXiv:2506.21495*, 2025.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024.

Jiawei Liu and Lingming Zhang. Code-r1: Reproducing r1 for code with reliable rewards. https://github.com/ganler/code-r1, 2025.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025a.

Zijun Liu, Peiyi Wang, Runxin Xu, Shirong Ma, Chong Ruan, Peng Li, Yang Liu, and Yu Wu. Inference-time scaling for generalist reward modeling. *arXiv preprint arXiv:2504.02495*, 2025b.

Saüc Abadal Lloret, Shehzaad Dhuliawala, Keerthiram Murugesan, and Mrinmaya Sachan. Towards aligning language models with textual feedback. *arXiv preprint arXiv:2407.16970*, 2024.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl, 2025. Notion Blog.

Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun Ma, and Wenhu Chen. General-reasoner: Advancing llm reasoning across all domains. *arXiv preprint arXiv:2505.14652*, 2025.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Gaurav Pandey, Yatin Nandwani, Tahira Naseem, Mayank Mishra, Guangxuan Xu, Dinesh Raghu, Sachindra Joshi, Asim Munawar, and Ramón Fernandez Astudillo. Brain: Bayesian reward-conditioned amortized inference for natural language generation from feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 39400–39415, 2024.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Penghui Qi, Zichen Liu, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Optimizing anytime reasoning via budget relative policy optimization. *arXiv preprint arXiv:2505.13438*, 2025.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Alireza Salemi and Hamed Zamani. Learning from natural language feedback for personalized question answering. *arXiv preprint arXiv:2508.10695*, 2025.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Joar Max Viktor Skalse and Alessandro Abate. The reward hypothesis is false. In *NeurIPS ML Safety Workshop*, 2022.

Moritz Stephan, Alexander Khazatsky, Eric Mitchell, Annie S Chen, Sheryl Hsu, Archit Sharma, and Chelsea Finn. Rlvf: Learning from verbal feedback without overgeneralization. *arXiv preprint arXiv:2402.10893*, 2024.

Richard Sutton. The reward hypothesis. http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html, 2004.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Yubo Wang, Xiang Yue, and Wenhu Chen. Critique fine-tuning: Learning to critique is more effective than learning to imitate. *arXiv preprint arXiv:2501.17703*, 2025a.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, et al. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025b.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I Wang. Swe-rl: Advancing llm reasoning via reinforcement learning on open software evolution. *arXiv preprint arXiv:2502.18449*, 2025.

Liang Wen, Yunke Cai, Fenrui Xiao, Xin He, Qi An, Zhenyu Duan, Yimin Du, Junchen Liu, Lifu Tang, Xiaowei Lv, et al. Light-r1: Curriculum sft, dpo and rl for long cot from scratch and beyond. *arXiv preprint arXiv:2503.10460*, 2025.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning. *arXiv preprint arXiv:2505.10320*, 2025.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2502.14768*, 2025.

Zhenghai Xue, Longtao Zheng, Qian Liu, Yingru Li, Xiaosen Zheng, Zejun Ma, and Bo An. Simpletir: End-to-end reinforcement learning for multi-turn tool-integrated reasoning. *arXiv preprint arXiv:2509.02479*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

John Yang, Kilian Lieret, Carlos E Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. Swe-smith: Scaling data for software engineering agents. *arXiv preprint arXiv:2504.21798*, 2025.

Shunyu Yao. The second half. https://ysymyth.github.io/The-Second-Half/, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Xiang Yue, Tianyu Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 37:90629–90660, 2024.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, et al. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*, 2025.

Hongyu Zang, Xin Li, Jie Yu, Chen Liu, Riashat Islam, Remi Tachet Des Combes, and Romain Laroche. Behavior prior representation learning for offline reinforcement learning. *arXiv preprint arXiv:2211.00863*, 2022.

Weihao Zeng, Yuzhen Huang, Wei Liu, Keqing He, Qian Liu, Zejun Ma, and Junxian He. 7b model and 8k examples: Emerging reasoning with reinforcement learning is both effective and efficient. https://hkust-nlp.notion.site/simplerl-reason, 2025. Notion Blog.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Xiaoying Zhang, Hao Sun, Yipeng Zhang, Kaituo Feng, Chaochao Lu, Chao Yang, and Helen Meng. Critique-grpo: Advancing llm reasoning with natural language and numerical feedback. *arXiv preprint arXiv:2506.03106*, 2025.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Xiangxin Zhou, Zichen Liu, Anya Sims, Haonan Wang, Tianyu Pang, Chongxuan Li, Liang Wang, Min Lin, and Chao Du. Reinforcing general reasoning without verifiers. *arXiv preprint arXiv:2505.21493*, 2025a.

Yang Zhou, Sunzhu Li, Shunyu Liu, Wenkai Fang, Jiale Zhao, Jingwen Yang, Jianwei Lv, Kongcheng Zhang, Yihe Zhou, Hengtong Lu, et al. Breaking the exploration bottleneck: Rubric-scaffolded reinforcement learning for general llm reasoning. *arXiv preprint arXiv:2508.16949*, 2025b.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

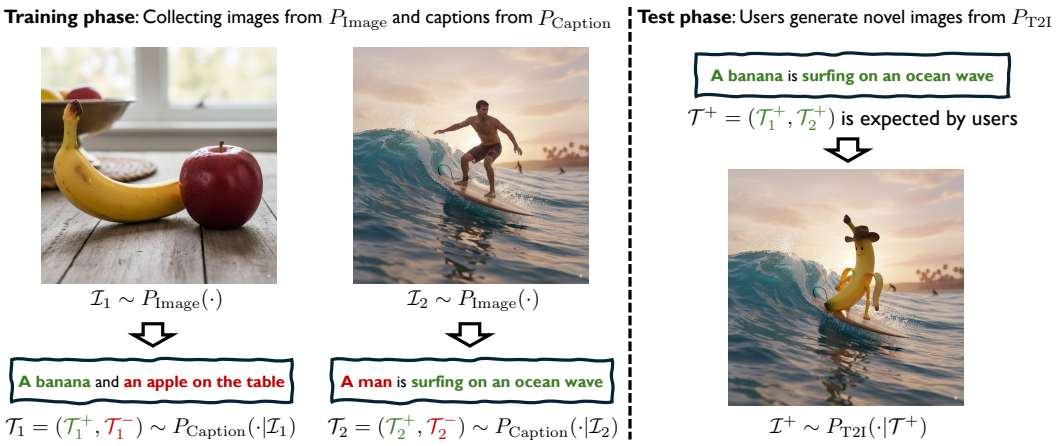


Figure 4: **Learning from mixed captions in text-to-image generation.** During training, models learn from realistic image-caption pairs such as "*a banana and an apple on the table*" or "*a man surfing on an ocean wave*". They can leverage language priors to recombine these captions and generate novel concepts, such as "*a banana surfing on the ocean*" (images shown are generated with Gemini 2.5 Flash Image). By analogy to Figure 1, this illustrates how diverse verbal feedback can be treated as a conditioning signal, motivating our feedback-conditional learning paradigm.

A Additional Derivations and Discussions

A.1 Proof of Eq. (2) and its special case

Following Rafailov et al. (2023), the optimal solution to a KL-constrained reward maximization problem $\mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})\right] - \mathbb{D}_{\text{KL}}\left(\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)||\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\right)$ can be written as

$$\begin{aligned}\pi^*(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+) &= \frac{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot \exp\left(\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})\right)}{\sum_{\boldsymbol{o}} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot \exp\left(\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})\right)} \\ &= \frac{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})}{\sum_{\boldsymbol{o}} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})} \\ &= \frac{P_{\text{off}}(\boldsymbol{o},\boldsymbol{c}^+|\boldsymbol{x})}{P_{\text{off}}(\boldsymbol{c}^+|\boldsymbol{x})} = P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+).\end{aligned} \tag{6}$$

Note that the objective in Eq. (2) is equivalent to minimizing the *reverse* KL divergence between $\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$ and $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$:

$$\begin{aligned}&\mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\log p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})\right] - \mathbb{D}_{\text{KL}}\left(\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)||\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})\right) \\ =\ &-\mathbb{D}_{\text{KL}}\left(\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)||P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)\right) + \log P_{\text{off}}(\boldsymbol{c}^+|\boldsymbol{x}).\end{aligned} \tag{7}$$

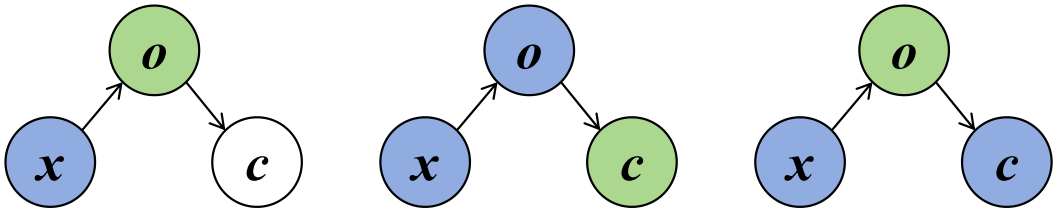
In the special case where the environment provides *verifiable rewards*, that is, $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o}^+) = 1$ for correct responses $\boldsymbol{o}^+$ and $p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o}^-) = 0$ for incorrect responses $\boldsymbol{o}^-$, we can show that $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$ reduces to the optimal solution of a 0-1 reward maximization problem without KL regularization: $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+) \in \arg\max_\pi \mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right]$. Specially, we have

$$\begin{aligned}P_{\text{off}}(\boldsymbol{o}^+|\boldsymbol{x},\boldsymbol{c}^+) &= \frac{\pi_{\text{ref}}(\boldsymbol{o}^+|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o}^+)}{\sum_{\boldsymbol{o}} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})} = \frac{\pi_{\text{ref}}(\boldsymbol{o}^+|\boldsymbol{x})}{\sum_{\boldsymbol{o} \text{ is } \boldsymbol{o}^+} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})}; \\ P_{\text{off}}(\boldsymbol{o}^-|\boldsymbol{x},\boldsymbol{c}^+) &= \frac{\pi_{\text{ref}}(\boldsymbol{o}^-|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o}^-)}{\sum_{\boldsymbol{o}} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot p_{\text{env}}(\boldsymbol{c}^+|\boldsymbol{x},\boldsymbol{o})} = 0.\end{aligned} \tag{8}$$

Thus, taking $\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+) = P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$ into the formula of $\mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right]$, we have

$$\mathbb{E}_{P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right] = \sum_{\boldsymbol{o} \text{ is } \boldsymbol{o}^+} \frac{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x}) \cdot \mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)}{\sum_{\boldsymbol{o} \text{ is } \boldsymbol{o}^+} \pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})} = 1. \tag{9}$$

Since there is $\max_\pi \mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right] = 1$, we know that $\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+) = P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)$ is one of the optimal solutions (not unique), i.e., $P_{\text{off}}(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+) \in \arg\max_\pi \mathbb{E}_{\pi(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c}^+)}\left[\mathbb{1}(\boldsymbol{o} \text{ is } \boldsymbol{o}^+)\right]$. □



(a) SFT (*behavior cloning*) (b) CFT (*forward dynamics*) (c) **Our FCP** (*inverse dynamics*)

Figure 5: **Graphical models for SFT, CFT, and our FCP.** Following Brandfonbrener et al. (2023), we use blue color to indicate inputs to the algorithm and green color to indicate prediction targets.

A.2 Connection to inverse dynamics modeling

In traditional RL, objectives for representation learning are often grouped into three classes: **behavior cloning**, **forward dynamics**, and **inverse dynamics**. Behavior cloning is typically used for imitation learning (Arora et al., 2020; Zang et al., 2022), forward dynamics is central to world modeling (Ha & Schmidhuber, 2018; Schwarzer et al., 2021), and inverse dynamics has been explored for both pretraining (Brandfonbrener et al., 2023) and feature extraction for exploration in RL (Efroni et al., 2021).

Interestingly, analogous structures appear in the LLM literature. The objectives of supervised finetuning (SFT), critique finetuning (CFT) (Wang et al., 2025a), and our feedback-conditional policy (FCP) align naturally with behavior cloning, forward dynamics, and inverse dynamics, respectively:

$$\begin{aligned}
\textbf{SFT (behavior cloning):} \quad & \max_{\pi_\theta} \mathbb{E}_{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})} \left[\log \pi_\theta(\boldsymbol{o}|\boldsymbol{x})\right]; \\
\textbf{CFT (forward dynamics):} \quad & \max_{\pi_\theta} \mathbb{E}_{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})} \left[\mathbb{E}_{p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})} \left[\log \pi_\theta(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})\right]\right]; \\
\textbf{Our FCP (inverse dynamics):} \quad & \max_{\pi_\theta} \mathbb{E}_{\pi_{\text{ref}}(\boldsymbol{o}|\boldsymbol{x})} \left[\mathbb{E}_{p_{\text{env}}(\boldsymbol{c}|\boldsymbol{x},\boldsymbol{o})} \left[\log \pi_\theta(\boldsymbol{o}|\boldsymbol{x},\boldsymbol{c})\right]\right].
\end{aligned} \tag{10}$$

We further illustrate this categorization with graphical models in Figure 5. This unified perspective clarifies the conditional structure underlying each finetuning paradigm and highlights how different forms of supervision drive model learning. In particular, our FCP extends the analogy by treating verbal feedback as a first-class supervision signal, positioning it as the natural *inverse-dynamics* counterpart to existing finetuning objectives.

B Detailed experimental setup

All implementations are based on `llama-factory` (Zheng et al., 2024) and `verl` (Sheng et al., 2025). Hyperparameter settings for both offline and online stages of FCP are listed in Table 6.

For the two special tokens `<EF>` and `</EF>`, embeddings are initialized by sampling from a multivariate normal distribution with mean and covariance computed over existing token embeddings. For general reasoning bootstrapping, we adopt a fully online setup with batch size of 256, differing from the math setting to illustrate that FCP remains effective under both training strategies.

Finally, Figure 6 shows the unified prompt template used to elicit feedback from `GPT-5-nano`. The template produces three outputs in one response: a low-quality *real-world user*-style feedback, a high-quality *professional reviewer*-style feedback, and a scalar score summarizing overall quality.

C More Experiment results

D LLM usage

We used an OpenAI LLM (GPT-5) as a writing and formatting assistant. In particular, it helped refine grammar and phrasing, improve clarity, and suggest edits to figure/table captions and layout (e.g., column alignment, caption length, placement). The LLM did not contribute to research ideation, experimental design, implementation, data analysis, or technical content beyond surface-level edits. All outputs were reviewed and edited by the authors, who take full responsibility for the final text and visuals.

Table 6: **Hyperparameters for FCP training** used in the offline and bootstrapping (online) stages.

Hyperparameter	Offline	Online
learning rate	5e-6	1e-6
lr scheduler	cosine	constant
weight decay	0	0.01
warmup ratio	0.1	0
train batch size	512	2048
ppo mini-batch size	—	512
temperature	—	1.0
top_p	—	1.0
rollout_n	—	4
epoch	1	
max response length	4096	
loss type	cross-entropy loss	
loss aggregation mode	token-mean	
feedback environment	`GPT-5-nano`	
feedback style	*professional reviewer*	

Table 7: Performance comparison under verifiable (rule-based) and LLM-generated supervision across multiple training methods.

Method	Source	AIME24	AIME25	MATH500	Minerva	Olympiad	Avg.
Base	-	7.5	6.7	63.8	28.3	28.6	27.0
Base + GRPO	rule-based verifier	13.3	14.2	76.3	36.6	41.7	36.4
Base + GRPO	LLM	20.0	13.3	75.7	42.3	40.8	38.4
RFT + GRPO	rule-based verifier	17.5	15.0	77.0	38.4	41.3	37.8
RFT + GRPO	LLM	25.8	9.2	75.1	36.8	38.9	37.1
FCP + Bootstrap	LLM	25.0	7.5	76.5	45.8	38.8	38.7

Table 8: Evaluation Results Under In-Distribution (ID) and Out-of-Distribution (OOD) Feedback Conditions

Method	Feedback Style	ID/OOD	AIME24	AIME25	MATH500	Minerva	Olympiad	Avg.
Base	-	-	7.5	6.7	63.8	28.3	28.6	27.0
FCP + Bootstrap	Reviewer	ID	25.0	7.5	76.5	45.8	38.8	38.7
FCP + Bootstrap	User	OOD	10.8	7.5	75.1	35.2	38.0	33.3
FCP + Bootstrap	"Correct"	OOD	16.7	7.5	75.3	35.2	37.2	34.4
FCP + Bootstrap	No Feedback	OOD	15.8	10.0	74.4	35.6	37.6	34.7

Table 9: Effects of training-used feedback quality on FCP performance across reasoning benchmarks.

Method	Feedback Style	Quality	AIME24	AIME25	MATH500	Minerva	Olympiad	Avg.
Base	-	-	7.5 ± 1.7	6.7 ± 0.0	63.8 ± 63.8	28.3 ± 0.8	28.6 ± 0.4	27.0 ± 0.5
FCP + Bootstrap	Correctness-Only	very low	10.0 ± 2.7	5.0 ± 1.9	73.4 ± 0.7	34.3 ± 0.6	35.3 ± 0.4	31.6 ± 0.8
FCP + Bootstrap	User	low	16.7 ± 2.7	0.8 ± 1.7	72.2 ± 0.4	37.1 ± 0.4	37.1 ± 0.8	32.8 ± 0.8
FCP + Bootstrap	Reviewer-Lite	medium	14.2 ± 4.2	8.3 ± 1.9	74.0 ± 0.3	37.4 ± 1.5	37.3 ± 0.7	34.2 ± 1.0
FCP + Bootstrap	Reviewer	high	25.0 ± 3.3	7.5 ± 1.7	76.5 ± 0.7	45.8 ± 0.7	38.8 ± 0.6	38.7 ± 0.7

Table 10: **Comparison of FCP performance using weak critique models.**

Method / Model	Critique Model	Feedback Style	AIME24	AIME25	MATH500	Minerva	Olympiad	Avg.
Qwen2.5-1.5B-Instruct	-	-	0.0	0.0	55.0	20.5	20.3	19.2
Qwen2.5-7B-Base	-	-	7.5	6.7	63.8	28.3	28.6	27.0
w/ FCP + Bootstrap	Qwen2.5-1.5B-Instruct	User	14.2	5.8	72.9	35.5	37.4	33.1
w/ FCP + Bootstrap	GPT-5-nano	User	16.7	0.8	72.2	37.1	37.1	32.8
w/ FCP + Bootstrap	GPT-5-nano	Reviewer	25.0	7.5	76.5	45.8	38.8	38.7

You are acting as a real-world human user of an LLM.

Inputs:
Question:
\"\"\"
{question}
\"\"\"

Model Answer:
\"\"\"
{model_answer}
\"\"\"

Reference Final Answer (used only for correctness check):
\"\"\"
{reference_answer}
\"\"\"

Your tasks:

1) Simulate "user feedback" from a normal, real-world user reacting to the Model Answer only.
- Length: 1-3 sentences, colloquial tone, first person.
- Content: purely subjective sentiment (e.g., helpfulness, confidence, confusion, satisfaction).
- STRICT: Do NOT mention or allude to any symbols, formulas, variable names, or specialized concepts from the Question or the Model Answer. Do NOT quote text from the inputs.

For example:
"I think you are right, but your solution is really long and complicated."
"You are a genius! You have all my respect."
"I am confused. There seems to be a mistake in your solution."
"What are you talking about? You are not answering my question."
etc.

2) Simulate a professional reviewer evaluating the Model Answer along several dimensions, including but not limited to:
• correctness — Compare the Model Answer's final result ONLY against the Reference Final Answer (if provided). Judge whether the end result matches; do not use the reference for any other purpose.
• logical_rigor — Assess the soundness and gaplessness of reasoning within the Model Answer itself. Do NOT use the Reference Final Answer here.
• completeness — Judge coverage of required parts and edge cases based on the Question and the Model Answer only. Do NOT use the Reference Final Answer here.
• clarity — Evaluate organization, readability, and ease of following in the Model Answer. Do NOT use the Reference Final Answer here.

Then provide a high-level summary (1-3 sentences) with overall judgment and broad observations.
- STRICT for the high-level summary: Only use adjectives and adverbs to describe the Model Answer and reasoning process. DO NOT mention where it goes wrong and where it can do better.

For example:
"Your final answer is correct, but the solution is too long and complicated. There are also several logical errors in your solution."
"The answer is partially correct. The reasoning is sound but not complete. Also, you are being too verbose."
"The answer is totally wrong. It lacks soundness and is not complete. However, the solution is concise and clear."

Hard constraints:
- Keep all content in English.
- Do not mention anything like "reference" or "python snippet".

Output format:
User-style feedback: <your 1-3 sentence feedback>
Analysis along several dimensions: <your 1-3 sentence analysis>
High-level summary: <your 1-3 sentence summary>
Score (0-10): <one overall integer score>

Figure 6: Prompt template used to elicit feedback from `GPT-5-nano`, including *real-world user*-style feedback, *professional reviewer*-style feedback, and a scalar score.